ESD-TR-71-236

MTR-2091

THE AIR FORCE JOVIAL COMPILER VALIDATION SYSTEM (JCVS)

F. Engel, Jr.

AUGUST 1971

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 8510

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts
Contract F19(628)-71-C-0002

Approved for public release; distribution unlimited.

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy

ESD-TR-71-236 MTR-2091

THE AIR FORCE JOVIAL COMPILER VALIDATION SYSTEM (JCVS)

F. Engel, Jr.

AUGUST 1971

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 8510
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract F19(628)-71-C-0002

FOREWORD

This report has been prepared by The MITRE Corporation under Project 8510 of Contract F19(628)-71-C-0002. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, L. G. Hanscom Field, Bedford, Massachusetts.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

ROBERT F. JENSEN, Colonel, USAF
Director of ADPE Selection
Deputy for Command and Management Systems

ABSTRACT

The Air Force JOVIAL Compiler Validation System (JCVS) was developed to assist in the validation of performance of proposed JOVIAL language processors. This report discusses the JCVS in the context of its use by the Air Force Directorate of ADPE Selection. It provides a certification of the audit test modules, and makes recommendations for improvements to the JOVIAL audit capability. It is recommended that the audit programs be used independently of the JCV System.

ACKNOWLEDGMENTS

The investigations and computer experimentation which served as the basis for this report were carried out with the assistance of Miss Tobyanne Paster, D73. Mr. Richard Robinson of RADC ISIS assisted in making the runs on the GE-635 computer at RADC, and has cooperated in exchanging results of subsequent experiences with JCVS. Lt. Col. V. Godbey and Lt. R. Forney of MCS Software Technical Group also participated in this effort. While acknowledging their valuable assistance, the author accepts full responsibility for the statements and conclusions reported herein.

TABLE OF CONTENTS

LIST OF TABLES

SECTION I

INTRODUCTION

The Air Force procedure for the competitive procurement of general purpose automatic data processing equipment requires the validation of the performance claims made for each proposed ADP system in order to establish the responsiveness to the systems requirements of the Request for Proposal. Among these requirements, there frequently occurs the need for a JOVIAL language processor which must conform to the AFM 100-24 JOVIAL (J3) Language Specifications.(1) The Air force JOVIAL Compiler Validation System (JCVS) (2) was developed under contract to assist in the task of validation of the performance of proposed JOVIAL language processors. This report briefly discusses the JCVS in the context of its possiole use by the Air Force Directorate of ADPE Selection, MCS, summarizes the experience acquired in working with the JCVS, critiques the audit test modules, and makes recommendations for improvements to the JOVIAL audit capability.

The compiler validation process or audit function has two purposes: the first is to establish that each language feature of the Air Force standard programming language is accepted by the JOVIAL processor under examination; the second is to establish that the execution of each language feature produces the prescribed results. The JCVS accomplishes this audit function by presenting to the JOVIAL processor a set of one or more JOVIAL source programs and their required input data (if any), which together contain statements invoking each of the required standard JOVIAL features. Such programs are referred to as "Audit Programs." Upon successful compilation and execution of the audit programs, it is established that a processor does conform to the standard for those features tested. While it is impossible for the audit programs to check for the correct implementation of all possible combinations of standard features due to the large number of tests that would be required, it is expected that a sufficiently representative sample will be included to establish reasonable confidence in the capability of the observed system.

SECTION II

THE JCVS COMPONENTS

The heart of the Air Force JCVS is the Population File. It contains all of the JOVIAL source statements comprising the tests of the standard features and the auxiliary procedures for reporting the results of execution of the tests. In addition to the Population File, the JCVS consists of a Population File Maintenance program, a Selector program, a Source Program Maintenance program, and a conversion program for character set transformation. These auxiliary programs are written in ANS COBOL and provide the means for automating the manipulation of the constituent elements of the JOVIAL audit programs.

THE POPULATION FILE

The Population File (Pop File) is the data base for the JCVS. It contains approximately 9000 cards of JOVIAL source statements and commentary which are subdivided into test modules of, at most, 50 cards each. Each test module contains all of the JOVIAL statements necessary to effect the test of a given feature including all item declarations, procedure declarations, etc. The test module is completely self-contained, with the exception of the output procedures, which are invoked to record the results of the test. The report writing or output procedures are contained in a separate module, providing a uniform reporting mechanism. Thus, each test module as it exists in the Pop File may be compiled and executed independently of every other test module.

The test modules may also be compiled and executed together as a single JOVIAL program by combining the individual modules in any desired sequence. Through the use of appropriate conventions in assigning names to variables, labels, procedures, etc., duplication of names has been avoided. The characteristics of the J3 language are such that no rearrangement or sorting of the JOVIAL statements of the test modules is necessary.*

*In this regard the J3 language differs significantly from both FORTRAN and COBOL. In FORTRAN specification statements must precede all executable statements, and in COBOL appropriate statements must appear in the ENVIRONMENT, DATA and PROCEDURE divisions of the program in that order (cf MTR-1953, The Air Force COBOL Compiler Validation System (CCVS)).

The physical arrangement of the Pop File consists of 4000 characters, fixed length records, each containing the characters from fifty 80-character cards. The first card of each record is a header card which identifies the module and may also designate a dependence upon other specific modules. The header card is in the form of a JOVIAL comment line and is considered to be a part of the JOVIAL test module. If more than 50 cards are required for a test, there is provision on the header card to specify that the module is to be appended to the preceding module as a continuation of it. Finally, the header card may designate that the module is to be present in every audit program, whether or not its selection has been requested. Characters 73-76 of every card contain the module identification number, and characters 78-80 contain the card sequence number within the module, which must be in the range of 1 to 50 inclusive. The 77th character of the header card designates which of the foregoing functions is intended for that module. On all other cards, this character designates JOVIAL source statements. The module identification number, the card sequence number and the function character provide the key basis for the functional manipulations performed upon the Pop File by the other components of the JCV System.

The first record of the Pop File has the module identification number 001, and fulfills additional unique functions. The first five cards of that record may be used to convey to the user for informational purposes only data concerning the source of the particular Pop File and the hardware environment in which it may operate. The content of these cards is ignored by the JCVS. Succeeding cards in the first record are designated environmental software cards, and may be operating system control cards, JOVIAL processor control cards, etc., which are to be placed before and/or after every audit program when it is selected from the Pop File. By offsetting these environmental software cards by one character, the JCVS is able to handle operating system control cards which require special characters in Column 1.

THE POPULATION FILE MAINTENANCE PROGRAM

The Population File Maintenance Program (POPFM) provides a mechanism for creating a new Pop File either from cards for the individual modules, or by updating an old Pop File by means of deletion, replacement or insertion of either individual cards or entire modules on the basis of the module identification numbers and card sequence numbers. A report of the functions performed is generated by POPFM.

THE SELECTOR PROGRAM

The JCVS Selector Program (SELECT) operates on the Pop File to produce a single JOVIAL source program from one or more JOVIAL test modules residing on the Pop File. The user designates, by means of control cards, those modules which are to be selected. The selected modules appear in order by module number as they exist on the Pop File. The user may direct the selectrd modules to be written onto magnetic tape or punched cards. An audit file is produced by SELECT containing a record of the modules selected and additional messages relating to suspected error conditions encountered.

THE SOURCE PROGRAM MAINTENANCE PROGRAM

The JCVS Source Program Maintenance Program (SOPMM) operates on an existing JCVS Source Program File to update and generate a new Source Program File. The SOPMM provides the user with the ability to add information to the source program file, delete information from the source program file, or replace information on the source program file on a card image by card image basis, according to the module identification number and the card sequence number. The user may direct the new source program file either to magnetic tape or punched cards. An audit file is produced optionally by SOPMM providing a record of the diagnostic and trace messages as well as a source program listing when desired.

THE POPULATION FILE INITIATING PROGRAM

The JCVS Pop File Initiating Program (INIPOP) operates to initiate and, at the user's option, to re-number a Pop File either from an existing Pop File or from a card file containing the test modules. The module identification numbers only are re-sequenced as directed by user supplied control cards. Certain cross-references between modules are changed automatically when affected by the re-numbering process. An audit report is created which contains diagnostic messages, and an optional listing of JOVIAL source statements on the new Pop File. A punched card deck of the Pop File may be obtained as an optional output.

THE REPORT WRITER PROGRAM

The JCVS Report Writer Program (JCVSRP) operates on the Pop File to produce a listing of the test modules on the Pop File or a listing of all the test header cards on the Pop File, or both as directed by user supplied control cards.

The reader is referred to the JCVS User's Manual[2] for a more detailed description of the JCVS components and their use. In the work with the JCVS Test Modules which was performed on both the GE-635 at RADC and the IBM/360-50 at MITRE, it was found more convenient to manipulate the test module card decks manually, rather than to use the JCVS programs enumerated above. The JCVS programs require that control cards be prepared in addition to the JOVIAL statement modifications, their use requires additional computer runs to be made to generate the modified audit programs before they can be processed by the JOVIAL compiler, and the reruns occasioned by errors introduced in these peripheral activities lengthen the apparent turn around time to process an audit program with the JOVIAL compiler. However, the JCVS components have each been used at least once, and they appear to function as described. It was found necessary to introduce a few minor modifications to initialize print records and repair clerical errors.

SECTION III

THE JCVS POPULATION FILE

The present set of JCVS Test Modules contained in the Population File fall into four general categories:

1. The Pass/Fail Reporter and its verification tests
2. Tests of Declarations
3. Tests of Procedural Statements
4. Tests of Processing Declarations

Each of these categories of tests is discussed in this section with respect to its operational characteristics, the thorouhgness and rigor of test logic, and the completeness of coverage of the AFM 100-24 JOVIAL features.

PASS/FAIL REPORTER

The Pass/Fail Reporter consists of three JOVIAL Procedures contained in Test Module 9998. Their function is to print one line identifying the test module and a second line indicating that the test was successful or had failed. The Procedure OUTERR prints a line by calling a FORTRAN subroutine in which the information to be printed is the 40-character Hollerith argument. This FORTRAN subroutine is not provided, nor are any specifications for it given. It is not obvious how a standard conforming FORTRAN program could be made to manipulate successfully this Hollerith argument.

The procedure OUTERA is invoked at the beginning of execution of each test to set up the test module identification line. The 4-character Hollerith argument of this procedure is the identification number of the test, which is then combined with the invariant part of the message by use of the JOVIAL OVERLAY feature. The message is printed by invoking the procedure OUTERR.

At the conclusion of its execution, each test module invokes the procedure OUTERB to set up and print the Pass/Fail message. The argument of the procedure OUTERB is a 1-character Hollerith variable which is assumed to have been set to the value "Y" if the test were successful, and to "N" if the test failed. The EQ relational operator is used in an IF:clause to compare the argument with the Hollerith literal constant 1H(Y). If the condition is true, the 40-character Hollerith variable is assigned the value of the "test successful" message, and OUTERR is invoked to print it. If the condition is not true, then the variable is assigned the value of the "test failed" message and the procedure OUTERR is invoked by the same statement.

The Pass/Fail Reporter provides a uniform reporting mechanism for all tests, and localizes all output statements and interfaces with FORTRAN in this one module. This particular embodiment offers some disadvantages. The use of other than standard JOVIAL output procedures makes the Audit Programs processor dependent, so that they must be specifically adapted to each processor to be tested. The FORTRAN interface with a Hollerith argument introduces a dependency upon the processor under test. The Pass/Fail Reporter is complicated by the use of nested procedures, i.e. procedures invoking other procedures; by the use of argument passing between procedures; by the use of logic within the Reporter; and by the use of the OVERLAY feature. Thus, the Pass/Fail Reporter is more sophisticated and complex in terms of JOVIAL features employed than many of the tests which use it to report their success or failure, and might itself be a source of error, preventing the use of any of the audit programs. The use of the OVERLAY feature might introduce processor dependencies due to word boundaries in data which are to be written out. Finally, the Pass/Fail Reporter provides no information on the test results which can be used either to confirm the success or to assist in diagnosing the failure of a test.

The function of the five test modules 0500-0520 is to validate the Pass/Fail Reporter. As originally designed, these modules attempted to verify the correct functioning of a processor for each of the JOVIAL language features which are employed in the Reporter. To do this independently of the Reporter itself, non-standard JOVIAL output statements were employed to print the test results. These statements had been subsequently replaced with calls to the Pass/Fail Reporter procedures for printing, with the result that the desired independent verification of these features has not been achieved. A further defect in these tests is that the features presumed to be tested are not tested under the same conditions of use in the Pass/Fail Reporter. For example, the Pass/Fail Reporter uses an IF:clause having a Hollerith:literal:relation list as its boolean:formula, but in test module 0520 the IF:clause feature is tested using only the boolean:constants 0 and 1 for the boolean:formula. Certainly different machine code is required for the relational comparison of literal formulas than for boolean:constants, and the success of the test of the latter case does not imply that the processor will treat the former correctly. Further, a sophisticated optimizing processor might recognize the invariance of the boolean:constants and generate code which would ignore completely the conditional statement as employed in test module 0520, thus completely invalidating the purpose of the test.

The following JOVIAL features are those presumed to be tested by modules 0500-0520:

1. Hollerith Item Description
2. Preset Hollerith Item
3. Procedure Call using Hollerith Argument
4. Hollerith Assignment Statement
5. GOTO Statement
6. IF:clause, boolean:constant
7. Procedure Definition, with no Argument
8. Procedure Call, with no Argument

The Pass/Fail Reporter Module 9998 uses all of these features except the last two items, and in addition, employs the OVERLAY feature, the definition of a procedure having a Hollerith argument, and the nesting of procedures with argument passing required. These last features are not explicitly validated by test modules 0500-0520.

A few specific programming errors were noted in module 9998. The global variables OUT1, OUTA, OUTB were not defined. Since these variables are referenced in almost every module, it is necessary that their item:descriptions occur at the beginning of the audit programs. The argument AA of the procedure OUTERR(AA) was not passed to the output subroutine, so that the message to be printed was lost. Finally, in the definition of the procedure OUTERA(CC), its argument was illegally declared to OVERLAY another datum. Since at execution time the dummy argument CC is replaced by any number of actual arguments, each of which may be defined in a different calling program unit, such use of OVERLAY is inconsistent.

In view of the difficulties enumerated above, MITRE provided a new module 0198 to replace test module 9998 as the Pass/Fail Reporter. This module is listed in Appendix I. The procedure names and arguments have been retained to be compatible with the existing procedure calls, but each procedure is now independent of the others; i.e. there is no nesting of procedures and concommitant argument passing. Global variables are used to transmit information to the procedures rather than relying on the procedure arguments. The use of the OVERLAY feature was retained, but it now occurs at the global level, rather than within the output procedures, and it does not involve the procedure arguments. This could be avoided by using a larger variety of print files to accommodate the different message lengths, by making all data for output the same length, (e.g. OUTA, OUTB and OUT1), or by introducing a more complex feature such as an array or table for constructing the output message. Short of revising the entire mechanism, it was felt that this represents the best compromise. A further modification was made to cause the value of the Pass/Fail indicator to be printed as part of the Pass/Fail Message, and to reset this parameter to the failed condition, i.e. OUTA=1H(N), after each success or fail message is printed. As a result the next test module must set the parameter to "Y", otherwise the default condition would be that the test failed.

Also provided is a new test module 0100 to validate this Pass/Fail Reporter. Rather than attempt to validate independently each of the features employed therein, it assumes that all of the features are implemented correctly, and attempts to demonstrate that each of the three procedures does indeed produce the expected results when called with appropriate parameter values. One line is printed directly by a JOVIAL OUTPUT statement in this test module when the print file is opened. A second line is printed by a call to the procedure OUTERR, and a third line results from a call to the procedure OUTERA, which incorporates the four-character module number into the test header line. Then to test the Pass/Fail function, the procedure OUTERB is called, first with the parameter OUTA set to "N", then again after setting OUTA to "Y" to demonstrate that the appropriate pass and fail messages are printed in each instance. Then, OUTERB is called a third time to demonstrate that the parameter OUTA had been reset to "N" after printing the "test successful" message. The test module then prints a final line using a direct JOVIAL OUTPUT statement, indicating the end of the test and the expected number of lines which are printed. The auditor must examine the output and determine that the appropriate messages have been printed in the expected sequence.

The JOVIAL <u>file:declaration</u> which defines the print file is included in the test module 0100, as are also the <u>item:descriptions</u> for the global parameters used by the audit programs. It is therefore necessary that modules 0100 and 0198 must be present in every audit program, and that they should be positioned at the beginning of the program. Hence, the relatively low module numbers which have been given them. The AFM 100-24 provides that the <u>device:name</u> occurring in the <u>file:declaration</u> is to be defined by each implementor. It is therefore to be expected that this statement which occurs only once in module 0100 will have to be changed for each processor to be tested.

In testing the JCVS with the J3 processor on the IBM system/360 Model 50, the efficacy of the revised Pass/Fail Reporter and validation procedure was demonstrated, when it was found that the "test successful" message was always printed, regardless of whether the parameter OUTA had the value "Y" or "N". This was due to a processor deficiency, which would not have been discovered by the original verification procedure.

DECLARATIONS

There are 151 test modules which are designed to test the declaration features of the JOVIAL language. Each test references the appropriate section of AFM 100-24 which describes the feature being tested, and the set of data declarations for simple items, arrays and tables of all types appears to be reasonably complete with respect to the variety of permissible forms which are represented.

The item declarations are the means for associating names with elements of a program and for describing the attributes of such elements. Item declarations are usually non-executable; i.e. they do not of themselves result in the generation of executable machine instructions. They provide information to the processor useful at compile time only, affecting storage allocation and the form of executable machine instructions that are generated for other statements. For this reason, the set of JCVS tests which are designed to validate the item definition features, e.g. modules 1000-1235, are inconclusive in themselves as to what features are or are not implemented correctly. Each of these tests upon execution will always produce the "test successful" message, signifying only that the test module was included in the audit program. The message has no significance relative to the capability of the JOVIAL processor under test to accept and interpret the feature. Careful analysis is required on the part of the auditor to validate the capability of the processor to handle these features. The compile time processor output might be helpful to the auditor, but it, too, may be inconclusive. Error flags or diagnostic information, if present, indicate things the processor does not recognize as valid, but the absence of such commentary does not mean that the features have been correctly interpreted and implemented. In fact a poor diagnostic or error detection facility in the processor would produce the same results.

As a further specific example, there occurs in test modules 1200 and 1210 the use of the range declaration for integer and fixed type data. According to the language specifications, the processor makes use of this information to provide adequate scaling and precision of representation of <u>intermediate</u> results. In order to test this feature, it would be necessary to define data and formulas using them which would demonstrate that scaling of the intermediate results was correct. This has not been done in these or any other of the tests included in the Pop File.

In summary, these tests give no assurance that the processor has <u>interpreted</u> each feature correctly, since this can only be done by the execution of appropriate statements which reference the items and validate the results.

PROCEDURAL STATEMENTS

There are 162 test modules in the Pop File covering the features of the procedural statements in the JOVIAL language. These modules also reference the appropriate sections of AFM 100-24 which describe the features being tested, and the systematic approach which has been followed seems to have provided for the inclusion of all of the essential features in the tests, with the exception that tests of all Input/Output features have been purposely excluded, as have tests of the DIRECT:statement (i.e. in-line machine code). These are considered to be processor dependent features (cf.(2) pp. 1, 5).

In general these tests are simple and straightforward in design and adequately validate the acceptance of the features being tested. The inaccessability of the computed results and test values complicate the analysis of test failures. Some of the test modules make use of features other than those under test, so that a test failure might be attributed to the wrong cause, again complicating the auditor's analysis of the failure. While the JCVS design made provision for indicating interdependencies among the test modules on the test header card, we found no instance of its use to ensure that dependent features would be tested automatically.

The tests of arrays and tables generally use a small number of entries; most often 2 in the case of tables, and sometimes 1 for an array dimension. While these cases are consistent with the standard, a more comprehensive test of larger structures would be desirable to demonstrate the capability of the processor to handle such references. It is also to be noted that many of the tests (cf module 3655) rely on the successful referencing of a single element of an array or table for validation. Again, a more comprehensive testing of many element references would provide better assurance of the processor's capability.

A detailed analysis of one of the more complex test modules, module 5310, suggests several ways in which the tests might be improved from the standpoint of the audit function. Module 5310 tests the alternative:statement; i.e., the use of IFEITH...ORIF... The JOVIAL alternative:statement provides for the sequential testing of any number of conditions, each of which except the first occurs in a successive or:if:clause. Associated with each condition is a single independent statement which will be executed if and only if that condition is true and every condition preceding it is false. Following execution of the associated statement, control is to be returned to the point which would follow the last or:if:clause if none of the set of alternative conditions were true. Thus, for an n-condition alternative there are in general n+1 paths leading to the following statement.

In the test module 5310, there are eight alternative:statements, two of them having three alternative conditions, the others having just two alternative conditions in each. The expected normal execution of the program causes each alternative statement to be executed only once, so that the remaining two or three possible alternative paths are not tested for each statement. Instead, this test module demonstrates different types of branching; i.e. sequences of alternative conditions in different alternative:statements, which are themselves differently structured. Hence, it does not conclusively demonstrate that the processor does indeed correctly handle the set of alternatives. Several of the conditions in these alternative: statements are in the form of the boolean constants 0 or 1; i.e., they are invariant. A sophisticated JOVIAL processor with optimization might recognize these and generate different machine code, bypassing the tests that would be required for the general, variable alternatives. Thus, again, this method of testing does not provide assurance that the processor has correctly implemented or interpreted the feature. Finally, the logic of the test provides multiple paths to the successful conclusion of the test, so that a combination of errors in the interpretation of the alternative:statements could cause the test to appear to be successful. The points just enumerated reflect the primary emphasis of the JCVS tests on the demonstration of the capability of a processor to accept the standard features, and the lesser concern with the thorough demonstration of the correctness of interpretation or implementation of the features.

Modules 5311 and 5312 have been developed to illustrate the more thorough testing of the alternative:statement. In module 5311, a five condition alternative:statement is to be executed six times, exercising each possible branch. Upon completion a check is made to verify that each path had been traversed as expected. In module 5312, the same five conditions are employed, but internal statement labels and branching to them are added to demonstrate this additional capability required within an alternative:statement. A call to a procedure is also introduced to be executed as one of the branches. Listings of these two modules are included in the recommended changes in Appendix I.

PROCESSING DECLARATIONS

There are 43 test modules devoted to the JOVIAL processing: declarations. These features are switches and user defined closes, procedures and functions. Here, too, each test module references the appropriate section of AFM 100-24 describing the feature being tested. This set of tests seems adequate in terms of the variety of forms and features in this category which are included in the tests.

SECTION IV

VALIDATION OF THE JCVS TEST MODULES

In order to establish validity of the JCVS test modules for performing the audit functions, they were reviewed to ensure that:

1. They were free of mechanical and clerical errors;
2. They were free of syntactic errors, i.e. the JOVIAL source statements conformed to the syntactic forms of AFM 100-24;
3. They were free of logic errors; i.e. the results of each test were consistent with the requirements of AFM 100-24;
4. They tested a sufficient number and variety of the language features to provide an adequate sample of the AFM 100-24 requirements.

The adequacy of coverage of the JOVIAL language features is discussed in the previous section under the appropriate categories. The cross reference list* of features tested by the set of modules in the JCVS Pop File was compared to the features defined in AFM 100-24 on a paragraph-by-paragraph basis. No glaring omissions were discovered, and on a purely subjective basis it was concluded that the coverage was adequate for the intended purpose, with the reservations previously cited concerning the intentional omission of features from testing, the insufficiency of testing of alternatives and the lack of logical rigor of the tests. The set of tests is open-ended, and it is presumed that tests of the input/output features and of more combinations of features will be added in the future.

A two-phase procedure was followed in carrying out the error investigation. First, selected test modules were subjected to a desk review and analysis in which the source program listings were examined and checked for clerical errors and consistency with AFM 100-24. In some few instances flow charts were prepared to assist in the validation of the logic of the tests, as for example in the case of module 5310 cited above. Secondly, the test modules were subjected to machine processing on two different systems to provide an independent and more complete verification of the syntactic analysis and correctness of execution results for each test.

The GE-600 Line JOVIAL processor[4] was selected for testing the JCVS modules because of ready access to the system on the GE-635 at the Rome Air Development Center and because that processor was

*See reference 2, p. 196-225.

thought to conform closely to the AFM 100-24 language, the known exceptions being the Input/Output features, and the DUAL and STRING item: declarations. The entire JCVS Pop File was divided into arbitrary subsets each containing from 50 to 70 test modules from which were removed those modules requiring features known not to be implemented in the GE-600 Line JOVIAL processor. The subsetting was done to avoid exceeding the limitations imposed by the compiler, to increase the probability of successful compilation and execution of at least some of the test modules during the brief test period, and to enable analyses of the results of some runs to be made while others were in the processing queue. A copy of the Pass/Fail Reporter, module 9998, modified as described below, was incorporated into each subset to make each a separate audit program for independent compilation and execution. The set of job control cards shown in Table I was evolved through an initial series of eight runs, in which the principal difficulties encountered were in the setting of memory limits, the proper identification of Input/Output units and files, and in the interpretation of system error messages. These control cards were attached to each audit program deck to effect the desired processing. At least two runs were made with each subset with the exception of the subset containing modules 5190 through 6085. That subset was run only once, and the run aborted after execution of all but five of the test modules in the set.

After each run, the printed output was examined for error diagnostic indications which may have occurred during compilation. If the cause of the error could be determined quickly to be in the test module and an obvious correction could be made, this was done and the module was left in the audit program for re-processing. Otherwise, the test module in question was removed from the deck before another attempt at processing was made. It was observed that the GE-600 Line JOVIAL processor would ignore invalid statements during the code generation phase of compilation, and the program would execute with those parts missing. This would frequently result in the "Test Successful" message being printed for that test module when in fact the test had not been performed properly. Therefore, it was necessary to examine carefully each run to verify successful execution of the tests.

As a result of this processing, 86 JCVS test modules were compiled and executed successfully; 178 modules were compiled without generating any error indications but were not executed due to prior run termination; 46 test modules were found to contain problems requiring further analysis to determine the nature of the error and whether the processor or the test module was at variance with AFM 100-24; and 49 test modules were not processed. Modifications

Table I

GE-635 Job Control Cards for JOVIAL Compile and Execute Run

```
$       SNUMB   25460

$       IDENT   EMBIL,RROBINSON,JCVS  ,IH519,5581

$       OPTION  ERCNT/4500/,JOVIAL

$       JOVIAL  NDECK

$       LIMITS  08,32000,,10000

$       INCODE  IBMEL

  **  source deck goes here **

$       EXECUTE

$       ENDJOB
```

were made to ten of the test modules to achieve these results. These modifications are included in Appendix I, and the results are detailed in Table 2, Column 4.

The modified Pass/Fail Reporter module 9998 used the GE-600 Line JOVIAL output statements rather than requiring a FORTRAN subprogram for the output function because there was insufficient information to define the interface between the GE-600 Line JOVIAL and FORTRAN. While the GE-600 Line JOVIAL input/output features differ from those of the AFM 100-24 language, they are used here only to perform the reporting function, which usage does not interfere with the validity of the tests being reported. The specific modifications consisted of a file declaration statement and a file opening statement placed at the beginning of the audit program, and an output statement to replace the procedure call on line 9998J011, as follows:

```
FILE PRNT  V(NORM)  R06 $  "DECLARE PRINT FILE"
OUT(1,PRNT) $  "OPEN PRINT FILE"
I09998(3,PRNT,0,AA,7) $  "PRINT PASS/FAIL, ETC."
```

The IBM System/360 JOVIAL Compiler[3] was selected as the second machine processing test vehicle because it appeared to be the most complete implementation of the JOVIAL language and because it became available on the MITRE/360-50 system shortly after the work at RADC had demonstrated the usefulness of machine testing of the audit programs. The procedure followed was similar to that employed with the GE-635, except that only those test modules which had not executed successfully with the GE-600 Line JOVIAL were processed on the system/360. A new Pass/Fail Reporter module 0198 previously described was used in place of module 9998 in all /360 JOVIAL runs. When, after each run, examination of the printed output established the successful compilation and execution of a test module, that module was considered to be validated and was removed from further processing. Those test modules which either failed to execute successfully or had errors detected in compilation were analyzed further in detail. Corrections were made to sixteen modules which were found at variance with AFM 100-24. The corrected modules were processed again until they, too, were compiled, executed and verified successfully. As a result of this processing, 178 additional JCVS test modules were compiled and executed successfully. There remain 102 test modules whose status has not been resolved; i.e. it has not been established conclusively that these test modules are in conformance to AFM 100-24 requirements, although 31 of them did compile without error diagnostics. A complete summary of the validation results is presented in Appendix II.

The successful compilation and execution of the test modules when confirmed by examination of processor generated output, has been accepted as validation of the modules because it is unlikely that the same error in interpretation will be made in both the test module and the processor, and fail to be noted by the reviewer. On the other hand, the failure of a test module in either compilation or execution is not conclusive evidence that the test module is incorrect. It has been established that errors exist in both the GE-600 Line JOVIAL and the IBM/360 JOVIAL Processors. While it was beyond the scope of the present investigation to analyze processor failures and make the necessary corrections, it has been possible in a few instances to identify specific processor shortcomings, as for example, the IBM processor's failure to reference correctly a procedure argument in making a boolean comparison, and also the failure to branch correctly on an index switch. In these cases the generated code was at fault although the interpretation of the JOVIAL source language was correct. Aside from the 39 test modules involving dual items, there are 34 test modules using Table items which are in an unresolved status. It is suspected, but not verified, that the processor is responsible for these failures, and that the test modules are correct.

SECTION V

SUMMARY AND RECOMMENDATIONS

The primary objective of this study of the JOVIAL Compiler Validation System was to establish the validity of the set of tests with respect to their correctness and consistency with the AFM 100-24 Language Specification. Desk analysis and machine processing with the GE-600 Line JOVIAL and the IBM System/360 J3 JOVIAL verified that 260 of the JCVS test modules, including those corrected, meet the criteria for use in performance validation by MCS. An additional seventy-three test modules are satisfactory on the basis of analysis alone, but could not be verified by processing due to the absence of correct processor implementations of dual items and table features. There are thirty test modules which have not been validated by any means. They, too, might be used in the performance validation, with the qualification that they may be modified when and if discrepancies are established by processor rejection and subsequent analysis.

The recommended changes and corrections to the test modules to achieve these results are documented in Appendix I. They have been incorporated into the Population File, and the set of corrected test modules on both magnetic tape and punched cards has been provided to the Air Force Directorate of ADPE Selection, together with a printed listing of the source cards for the test modules.

The investigation of the JCVS test modules also established that there are areas in which improvements can be made with respect to the overall audit function. It is recommended that future work be undertaken 1) to improve the rigor and thoroughness of the tests, as exemplified by the MITRE developed modules 5311 and 5312, and 2) to provide the auditor with more useful information about the test results to aid in his analysis. These improvements would on the one hand provide further assurance of the correctness of processor implementation, and on the other enhance the usefulness of the JCVS in the MCS performance validation environment.

In the process of carrying out this study of the audit programs, the JCVS auxiliary programs were used on both the IBM/360 and GE-635 systems to prepare the punched card decks for processing. The use of the JCVS auxiliary programs was found to require more time and to be less convenient than manual preparation of the card decks. It is recommended that MCS distribute the JCVS audit programs on tapes directly to the vendors independently of the JCV System.

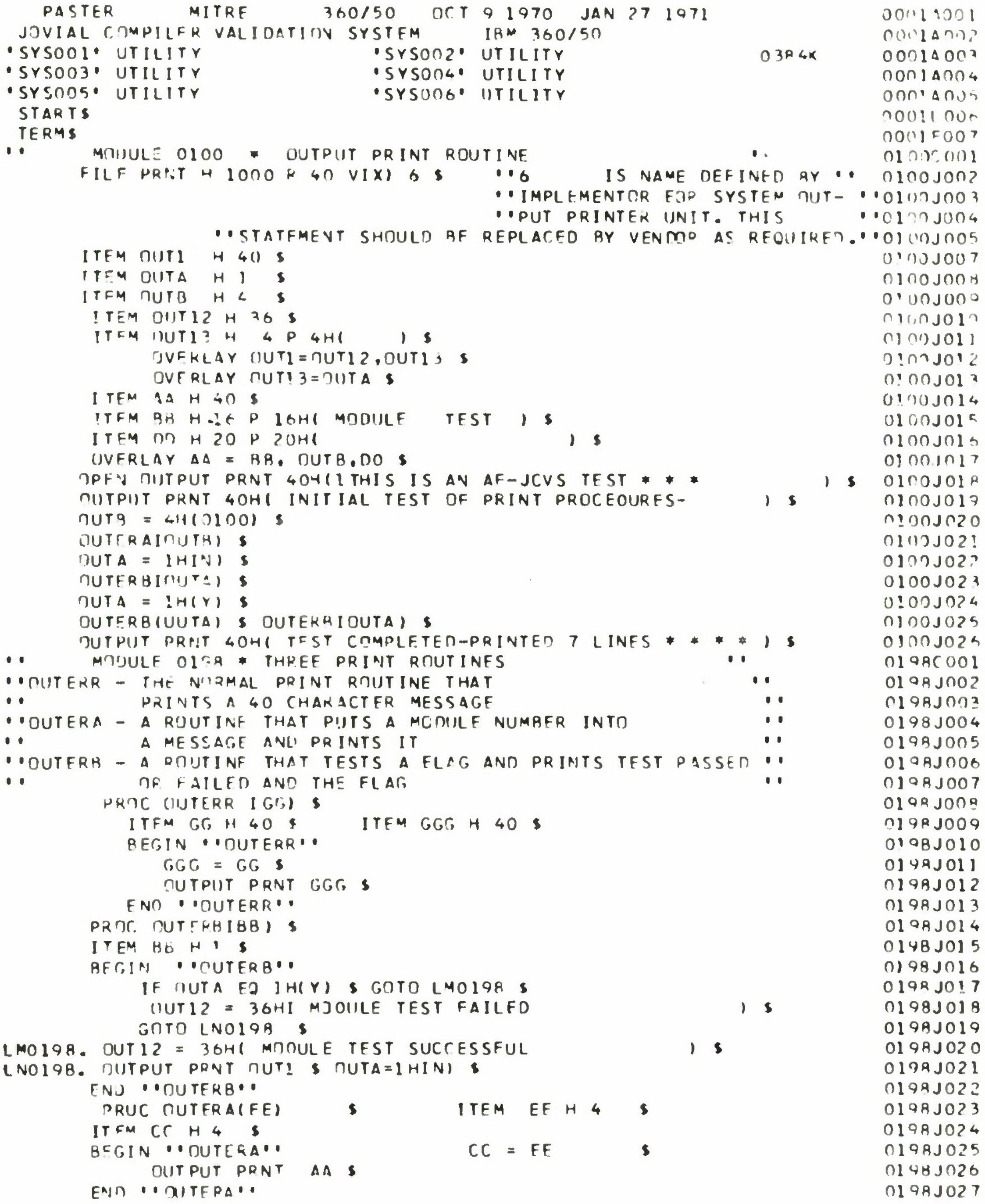
APPENDIX I

MODIFICATIONS TO THE JCVS TEST MODULES

The corrections to the JCVS test modules which were necessary to make them conform to the AFM 100-24 requirements, to correct clerical errors, or to improve the performance with respect to the audit function are cited in Table II. The format is that obtained by listing the JOVIAL source language cards for the changes. The four digits in columns 73-76 are the module numbers, and the last three digits give the card sequence numbers. These cards are to replace the same numbered cards in the Pop File. Also contained in this list are completely new modules 0100, 0198, 5311 and 5312 which are discussed in the text.

TABLE II

JCVS Test Module Corrections



```
   PASTER      MITRE      360/50   OCT 9 1970  JAN 27 1971                 0001A001
 JOVIAL COMPILER VALIDATION SYSTEM     IBM 360/50                          0001A002
'SYS001' UTILITY              'SYS002' UTILITY               03R4K         0001A003
'SYS003' UTILITY              'SYS004' UTILITY                             0001A004
'SYS005' UTILITY              'SYS006' UTILITY                             0001A005
 STARTS                                                                    0001L006
 TERMS                                                                     0001F007
''     MODULE 0100  *  OUTPUT PRINT ROUTINE                      ''        0100C001
      FILE PRNT H 1000 R 40 VIX) 6 $    ''6      IS NAME DEFINED BY ''     0100J002
                                        ''IMPLEMENTOR FOR SYSTEM OUT- ''0100J003
                                        ''PUT PRINTER UNIT. THIS      ''0100J004
                 ''STATEMENT SHOULD BE REPLACED BY VENDOR AS REQUIRED.''0100J005
      ITEM OUT1  H 40 $                                                    0100J007
      ITEM OUTA  H 1  $                                                    0100J008
      ITEM OUTB  H 4  $                                                    0100J009
       ITEM OUT12 H 36 $                                                   0100J010
       ITEM OUT13 H  4 P 4H(    ) $                                        0100J011
            OVERLAY OUT1=OUT12,OUT13 $                                     0100J012
            OVERLAY OUT13=OUTA $                                           0100J013
       ITEM AA H 40 $                                                      0100J014
       ITEM BB H 16 P 16H( MODULE   TEST  ) $                              0100J015
       ITEM DD H 20 P 20H(                    ) $                          0100J016
       OVERLAY AA = BB, OUTB,DD $                                          0100J017
      OPEN OUTPUT PRNT 40H(1THIS IS AN AF-JCVS TEST * * *         ) $      0100J018
      OUTPUT PRNT 40H( INITIAL TEST OF PRINT PROCEOURES-      ) $          0100J019
      OUTB = 4H(0100) $                                                    0100J020
      OUTERAIOUTB) $                                                       0100J021
      OUTA = 1HIN) $                                                       0100J022
      OUTERBIOUTA) $                                                       0100J023
      OUTA = 1H(Y) $                                                       0100J024
      OUTERB(UUTA) $ OUTERBIOUTA) $                                        0100J025
      OUTPUT PRNT 40H( TEST COMPLETED-PRINTED 7 LINES * * * * ) $          0100J026
''     MODULE 0198 * THREE PRINT ROUTINES                      ''          0198C001
''OUTERR - THE NORMAL PRINT ROUTINE THAT                        ''         0198J002
''          PRINTS A 40 CHARACTER MESSAGE                        ''        0198J003
''OUTERA - A ROUTINE THAT PUTS A MODULE NUMBER INTO              ''        0198J004
''          A MESSAGE AND PRINTS IT                              ''        0198J005
''OUTERB - A ROUTINE THAT TESTS A FLAG AND PRINTS TEST PASSED ''          0198J006
''          OR FAILED AND THE FLAG                               ''        0198J007
        PROC OUTERR IGG) $                                                 0198J008
          ITEM GG H 40 $       ITEM GGG H 40 $                             0198J009
          BEGIN ''OUTERR''                                                 0198J010
             GGG = GG $                                                    0198J011
             OUTPUT PRNT GGG $                                             0198J012
          END ''OUTERR''                                                   0198J013
       PROC OUTERBIBB) $                                                   0198J014
       ITEM BB H 1 $                                                       0198J015
       BEGIN  ''OUTERB''                                                   0198J016
           IF OUTA EQ 1H(Y) $ GOTO LM0198 $                                0198J017
            OUT12 = 36HI MJOULE TEST FAILED                   ) $          0198J018
           GOTO LN0198  $                                                  0198J019
LM0198. OUT12 = 36H( MODULE TEST SUCCESSFUL                ) $             0198J020
LN0198. OUTPUT PRNT OUT1 $ OUTA=1HIN) $                                    0198J021
       END ''OUTERB''                                                      0198J022
        PRUC OUTERA(EE)     $        ITEM  EE H 4   $                      0198J023
       ITEM CC H 4  $                                                      0198J024
       BEGIN ''OUTERA''              CC = EE        $                      0198J025
            OUTPUT PRNT  AA $                                              0198J026
       END ''OUTERA''                                                      0198J027
```

TABLE II (Continued)

```
''GOTO STAT-NAME        2444                                  ''       05I0A001
''PROCEDURE,ERR PRINT 2446  2484                              ''       0515A001
         ITEM IS0503 A 11 U   4 R  450...47500    $                    1010J029
         ITEM IT0503 A  5 U   7 P    0...1        $                    1010J030
         ITEM IU0503 A  7 J  -3 P 1000...175986   $                    1010J031
         ITEM IV0503 A 16 S  14 R   15...99E0     $                    1010J032
         ITEM IW0503 A 13 S   9 R 1400...175E1    $                    1010J033
         ITEM IX0503 A 15 S  11 R  675...850      $                    1010J034
         ITEM IM0521 I 10 S R  17...271 P   59 $                       1200J022
         IF IA1430 EQ V(ONI $ GOTO LY1430 $                            1430J009
         ORIF IA5210 EQ 1 $  GOTO LA5210 $                             1445J012
                  IFFITH SA5230 FO V(A) $ GOTO LZ5230$                 1455J011
                  ORIF SA5230 EQ V(A)$ GOTO LZ5230 $                   1455J017
         LB5230. ORIF SD5230 FQ V(YES)$ SC5230=V(NOI$                  I455J018
''THIS MODULE TESTS THE IFFITH,OPIF STATEMENTS                ''       1465J003
''MIXED DATA TYPES FOR BOOLEAN FORMALA                        ''       1465J004
         OUTB=4H(1465I $  OUTERA(OUTBI $                               1465H005
         ITEM IJ7155 H 4 P 4H(STEP)  $                                 1570J018
         TABLE TA7165 V 5 1 $                                          2500J007
         1 63                                                          2500J011
''CLASSIFICATION NUMBER  4.4.1.2                              ''       4122J002
                        IA7940($3$I=I$                                 4138J028
         IF5502($2$I= 3.14 $                                           4144J016
         IF5602($2$I= 3.14 $                                           4156J016
         IS5604($2$I= V(FFI $                                          4160J016
                 IF 'LOC(PA7500.)NO 1024$ GOTO LN7500$                 4175J008
                 GOTO LN7500$           ''ERROR''                      4175J010
LZ7540.  OUTA=1H(N)$                                                   4197J016
         10.A6 9.01625A6                                               4345J014
LB1200.  OUTFRB(OUTA)$                                                 4450J016
''PROCEDURES            2446  2484                            ''       4455A001
LPX125.                                                                4840J027
LD1200.  GOTO LC1200 $                                                 5160J009
''IFFITH, ORIF          2454                                  ''       5311A001
''CLASSIFICATION NUMBER  6.4.2                                ''       5311J002
''THIS MODULE TESTS THE ABILITY OF THE COMPILER TO            ''       5311J003
''USE THE IFFITH, ORIF STATEMENTS                             ''       5311J004
''THIS MODULE WRITTEN BY F. ENGEL, JR.   71.3.15   TEST!      ''       5311J005
         OUTB=4H(5311I $                                               5311H006
         OUTERA(OUTB) $                                                5311J007
         ITEM IA5311  I 16 S P O $                                     5311J008
         ITEM IB5311  I 16 S P O $                                     5311J009
         ITEM BA5311  B P O $                                          5311J010
LA5311. IFFITH BA5311 $ BEGIN IA5311=IA5311+16 $                       5311J011
                              BA5311=0 $  END                          5311J012
        ORIF IA5311 FO 1 $ IA5311=IA5311+2 $                           5311J013
        ORIF IB5311 GR 4 $ BEGIN                                       5311J014
                              IA5311=IA5311+8 $                        5311J015
                              BA5311=1 $                               5311J016
                              END                                      5311J017
        ORIF IA5311 FQ 3 $                                             5311J018
                              IA5311=IA5311+4 $                        5311J019
        ORIF IA5311 LS 7 $                                             5311J020
                              IA5311=IA5311+1 $                        5311J021
        END                                                            5311J022
        IF IB5311 LS 6 $                                               5311J023
             BEGIN  IB5311=IB5311+1 $                                  5311J024
                    GOTO LA5311 $  END                                 5311J025
        IF IB5311 FQ 6 $                                               5311J026
```

TABLE II (Concluded)

```
             BEGIN  IF IA5311 EQ 31 $                                          5311J027
                    GOTO LY5311 $  END                                         5311J028
        OUTA=1H(N) $                                                           5311J029
        GOTO LZ5311 $                                                          5311J030
LY5311. OUTA=1H(Y) $                                                           5311J031
LZ5311. OUTERB(OUTA) $                                                         5311J032
''IFEITH, ORIF          2454                                    ''             5312A001
''CLASSIFICATION NUMBER  6.4.2                                  ''             5312J002
''THIS MODULE TESTS THE ABILITY OF THE COMPILER TO              ''             5312J003
''USE THE IFEITH, ORIF STATEMENTS                               ''             5312J004
''THIS MODULE WRITTEN BY F. ENGEL, JR.   71.3.15   TEST2        ''             5312J005
         OUTB=4H(5312)$                                                        5312H006
         OUTERA(OUTB) $                                                        5312J007
         ITEM IA5312  I 16 S P 0 $                                             5312J008
         ITEM IB5312  I 16 S P 0 $                                             5312J009
         ITEM BA5312  B P 0 $                                                  5312J010
LA5312. IFEITH BA5312 $ BEGIN  IA5312=IA5312+16 $                              5312J011
                               BA5312=0 $  END                                 5312J012
LB5312. ORIF IA5312 EQ 1 $  GOTO LF5312 $                                      5312J013
        ORIF IB5312 EQ 2 $  PA5312 $                                           5312J014
LC5312. ORIF IA5312 EQ 3 $  BEGIN  IA5312=IA5312+4 $                           5312J015
                                   GOTO LB5312 $  END                          5312J016
        ORIF IA5312 LS 7 $  IA5312=IA5312+1 $                                  5312J017
        END                                                                    5312J018
LD5312. IF IB5312 EQ 4 $  GOTO LF5312 $                                        5312J019
        IF IB5312 GR 4 $  GOTO LX5312 $                                        5312J020
        IB5312=IB5312+1 $                                                      5312J021
        GOTO LA5312 $                                                          5312J022
LF5312. IA5312=IA5312+2 $                                                      5312J023
        GOTO LC5312 $                                                          5312J024
        PROC PA5312 $                                                          5312J025
           BEGIN ''PA5312''                                                    5312J026
             IA5312=IA5312+8 $                                                 5312J027
             BA5312=1 $                                                        5312J028
           END   ''PA5312''                                                    5312J029
LE5312. IF IA5312 EQ 31 $  GOTO LY5312 $                                       5312J030
LX5312. OUTA=1H(N) $                                                           5312J031
        GOTO LZ5312 $                                                          5312J032
LY5312. OUTA=1H(Y) $                                                           5312J033
LZ5312. OUTERB(OUTA) $                                                         5312J034
         PR3012 = 15.0A4 $                                                     5460J022
         OUTB=4H(6085)$                                                        6085H006
         OUTB=4H(6090)$                                                        6090H006
         OUTB=4H(6095)$                                                        6095H006
LE5340.  IF NOT (NOT(NOT BB5340)) $  GOTO LY5340 $                             6125J022
LY5340.                                                                        6125J025
''LOCATION                                                      ''             6735J005
         BEGIN ''TB7421''                                                      6745J017
         ITEM IJ7421 I 12 U $                                                  6745J018
LA7421.  IF IK7421($I$) EQ 12 $  GOTO LY7421 $                                 6745J025
```

APPENDIX II

RESULTS OF THE JCVS TEST MODULE VALIDATION

In Table III the entire set of JCVS Test Modules are listed by module number, indicating the results of machine processing and analysis for each module. The first column of this table indicates those modules which were modified by MITRE, with an asterisk designating those modules which were newly created by MITRE. In the next two columns are indicated the results of the machine processing on the GE-635 and IBM/360, respectively. The significance of the symbols appearing therein is as follows:

S - Successful compilation and execution without error
C - Compilation without error
M - Successful compilation and execution without error after modification by MITRE
F - Failed in execution after compilation without error
N - Errors detected during compilation
W - Withheld from processing for known deficiency in processor, e.g. dual item.

A mark in the fourth column indicates that by analysis of the source program and processing results, if any, the test module complies with the requirements of AFM 100-24. The last column of the table indicates those modules whose status is unresolved; i.e. there has been no confirmation of validity by successful machine processing of the module. A letter D in this column identifies test modules involving Dual Items, for which no processor implementation exists.

Table III

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
0100	P/F Reporter Validat	*		S		
0198	Pass/Fail Reporter	*		S		
0500	Define-Preset H Item		S		v	
0505	Assignment Statement		S		v	
0510	GOTO Statement		S		v	
0515	Procedure,Err Print		S		v	
0520	IF Clause		S		v	
1000	Simple Items		S		v	
1005	Simple Items		S		v	
1010	Simple Items	v	M		v	
1015	Simple Items		W		v	D
1020	Simple Items		S		v	
1025	Simple Items		S		v	
1030	Simple Items		S		v	
1035	Simple Items		S		v	
1200	Simple Items	v	C	M	v	
1205	Simple Items		C	S	v	
1210	Simple Items		C	S	v	
1215	Simple Items		W		v	D
1220	Simple Items		C	S	v	
1225	Simple Items		C	S	v	
1235	Simple Items		C	S	v	
1400	Simple Items		C	S	v	
1405	Simple Items		C	S	v	
1410	Simple Items		C	S	v	
1415	Simple Items		W		v	D
1420	Simple Items		C	S	v	
1425	Simple Items		C	S	v	
1430	Simple Items	v	C	M	v	
1435	Simple Items		C	S	v	

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
1440	Simple Items		C	F		v
1445	IFEITH, ORIF	v	C	M	v	
1450	IFEITH, ORIF		C	S	v	
1455	Simple Items	v	C	M	v	
1460	IFEITH, ORIF		C	F	v	v
1465	IFEITH, ORIF	v	N	MF	v	v
1500	Ordinary Tables		C	S		
1505	Ordinary Tables		N	F	v	v
1510	Ordinary Tables		W			D
1515	Ordinary Tables		C	S		
1520	Ordinary Tables		W			D
1525	Ordinary Tables		C	F		v
1530	Ordinary Tables		C	S		
1545	Ordinary Tables		C	S		
1550	Ordinary Tables		C	F		v
1555	Ordinary Tables		W			D
1560	Ordinary Tables		C	S		
1565	Ordinary Tables		W			D
1570	Ordinary Tables	v	N	MF	v	v
1575	Ordinary Tables		C	F		v
1580	Ordinary Tables		C	S		
1585	Ordinary Tables		W			D
1590	Ordinary Tables		C	S		
1595	Ordinary Tables		C	S		
1600	Ordinary Tables		W			D
1605	Ordinary Tables		C	S		
1610	Ordinary Tables		C	S		
1615	Ordinary Tables		C	S		
1620	Ordinary Tables		C	S		
1625	Ordinary Tables		C	S		

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
1630	Ordinary Tables		C	S		
1635	Ordinary Tables		C	S		
1700	Ordinary Tables		C	S		
1705	Ordinary Tables		C	S		
1710	Ordinary Tables		C	S		
1715	Ordinary Tables		C	S		
1720	Ordinary Tables		C	S		
1725	Ordinary Tables		C	S		
1730	Ordinary Tables		C	F		v
1735	Ordinary Tables		W			D
1740	Ordinary Tables		C	S		
1745	Ordinary Tables		C	S		
1750	Ordinary Tables		C	S		
1755	Ordinary Tables		C	S		
1760	Ordinary Tables		C	S		
1765	Ordinary Tables		C	S		
1770	Ordinary Tables		C	S		
1775	Ordinary Tables		W			D
1780	Ordinary Tables		C	S		
1785	Ordinary Tables		C	S		
1790	Ordinary Tables		C	S		
1795	Ordinary Tables		C	S		
1800	Ordinary Tables		C	S		
1805	Ordinary Tables		C	S		
1810	Ordinary Tables		C	S		

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
1815	Ordinary Tables		N	W		D
1820	Ordinary Tables		C	S		
1825	Ordinary Tables		C	S		
1830	Ordinary Tables		C	S		
1835	Ordinary Tables		C	S		
1840	Crdinary Tables		C	S		
1845	Ordinary Tables		C	S		
1850	Ordinary Tables		C	S		
1855	Ordinary Tables		C	F		v
1860	Ordinary Tables		W			D
1900	Ordinary Tables		C	S		
1905	Ordinary Tables		C	S		
1910	Ordinary Tables		C	S		
1915	Ordinary Tables		C	S		
1920	Ordinary Tables		C	S		
1925	Ordinary Tables		C	S		
1930	Ordinary Tables		C	S		
1935	Ordinary Tables		C	S		
1940	Ordinary Tables		C	S		
1945	Ordinary Tables		W			D
2500	Defined Tables	v	N	M	v	
2505	Defined Tables		N	F		v
2510	Defined Tables		N	N		v
2515	Defined Tables		N	N		v
2520	Defined Tables		N	F		v

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
2525	Defined Tables		C	S		
2530	Defined Tables		W			D
2535	Defined Tables		N	F		v
2540	Defined Tables		N	N		v
3500	Arrays		C	S		
3505	Arrays		C	S		
3510	Arrays		C	S		
3515	Arrays		C	S		
3520	Arrays		C	S		
3525	Arrays		C	N		v
3530	Arrays		W			D
3535	Arrays		C	S		
3540	Arrays		C	S		
3545	Arrays		F	F		v
3600	Defined Tables		W	N		v
3605	Defined Tables		W	N		v
3610	Defined Tables		W	N		v
3615	Defined Tables		W	N		v
3620	Defined Tables		W	N		v
3625	Defined Tables		W	N		v
3630	Defined Tables		W	S		
3635	Defined Tables		W			D
3640	Defined Tables		W	N		v
3645	Defined Tables		W	N		v
3650	Defined Tables		W	N		v

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
3655	Defined Tables		W	N		v
3660	Defined Tables		W	N		v
3990	Switches		S			
3905	Switches		S			
3910	Switches		S			
3915	Switches		F	F	v	v
3920	Switches		F	S		
3925	Switches		W			D
3930	Switches		S			
3935	Switches		S			
3940	Switches		S			
3945	Switches		S			
3950	Switches		S			
3955	Switches		S			
3960	Switches		S			
3965	Switches		N	F	v	v
4000	BIT		C	S		
4005	BIT		C	S		
4010	BIT		C	S		
4015	BIT		C	S		
4080	BYTE		C	S		
4085	BYTE		C	S		
4090	BYTE		C	S		
4095	BYTE		C	S		
4120	ENTRY and ENT		C	F		v

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
4122	ENTRY and ENT	v	N	MF		v
4124	ENTRY and ENT		C	N		v
4126	Entry and Ent		C	F		v
4128	Entry and Ent		C	F		v
4130	Entry and Ent		C	F		v
4132	Entry and Ent		C	F		v
4134	Entry and Ent		C	F		v
4136	Entry and Ent			F		v
4138	Entry and Ent	v	F	MF	v	v
4140	Entry and Ent		C	S		
4142	Entry and Ent		C	S		
4144	Entry and Ent	v	N	M	v	
4146	Entry and Ent		C	S		
4148	Entry and Ent		C	S		
4150	Entry and Ent		C	S		
4152	Entry and Ent		C	S		
4154	Entry and Ent		C	S		
4156	Entry and Ent	v	N	M	v	
4158	Entry and Ent		C	S		
4160	Entry and Ent	v	N	M	v	
4162	ENTRY and ENT		C	S		
4175	PRIME LOC		N	N		v
4178	PRIME LOC	v	C	NM	v	v
4181	PRIME LOC		N	N		v
4184	PRIME LOC		C	N		v
4187	PRIME LOC	v	N	M	v	
4190	PRIME LOC		S			
4193	PRIME LOC		F	N		v
4196	PRIME LOC		N	N		v
4199	PRIME LOC		N	N		v

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
4200	NENT		S			
4205	NENT		S			
4210	NENT		S			
4215	NENT		S			
4220	NENT		S			
4225	NENT		S			
4230	NENT		S			
4235	NENT		N	W		D
4300	NWDSEN		S			
4305	NWDSEN		S			
4310	NWDSEN		S			
4315	NWDSEN		S			
4340	ODD		S			
4345	ODD	v	F	M	v	
4350	ODD		S			
4390	ALL		S			
4450	Procedures	v	N	M	v	
4455	Procedures		C	S		
4460	Procedures		C	S		
4465	Procedures		C	S		
4470	Procedures		C	S		
4475	Procedures		C	S		
4480	Procedures		N	S		
4485	Procedures		N	S		
4490	Procedures		C	S		
4495	Procedures		C	S		
4470	Procedures		C	S	v	
4475	Procedures		C	S	v	
4780	Procedures		C	S	v	
4785	Procedures		C	S	v	

Table III (Continued)

JCVS Test Modules Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
4800	Functions		C	S		
4805	Functions		C	S		
4810	Functions		C	S		
4815	Functions		C	S		
4820	Functions		C	S		
4825	Functions		N	N		
4830	Functions		N	S		
4835	Functions		C	S		
4840	Functions	v	C	M	v	
4845	Functions		W			D
5100	Functions		C	S	v	
5105	Functions		C	S	v	
5110	Functions		C	S	v	
5115	Functions		C	S	v	
5120	Functions		C	S	v	
5160	Functions	v	N	M	v	
5180	RETURN		S			
5185	RETURN		S			
5190	RETURN		S			
5280	IF Clause		S			
5310	IFEITH, ORIF		S		v	
5311	IFEITH, ORIF	*		S	v	
5312	IFEITH, ORIF	*		S	v	
5340	FOR Loops		S			
5342	FOR Loops		S			
5344	FOR Loops		S			
5346	FOR Loops		S			
5348	FOR Loops		S			
5350	FOR Loops		S			
5352	FOR Loops		S			

Table III (Continued)

JCVS Test Module Valiation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
5354	FOR Loops		S			
5356	FOR Loops		S			
5358	FOR Loops		S			
5360	FOR Loops		S			
5362	FOR Loops		S			
5364	FOR Loops		S			
5366	FOR Loops		S			
5368	FOR Loops		S			
5370	FOR Loops		S			
5372	FOR Loops		S			
5390	Loop Control		S			
5392	Loop Control		S			
5394	Loop Control		S			
5396	Loop Control		S			
5400	Numeric Expression		S			
5405	Numeric Expression		F	F		v
5410	Numeric Expression		S			
5415	Numeric Expression		S			
5420	Numeric Expression		S			
5425	Numeric Expression		S			
5430	Numeric Expression		S			
5435	Numeric Expression		S			
5440	Numeric Expression		S			
5445	Numeric Expression		S			
5450	Numeric Expression		S			
5455	Numeric Expression		S			
5460	Numeric Expression		F	N		v
5465	Numeric Expression		S			
5470	Numeric Expression		F	S		
5475	Numeric Expression		S			

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
5480	Numeric Expressions		S			
5485	Numeric Expressions		W			D
5490	Numeric Expressions		W			D
5495	Numeric Expressions		W			D
5500	Numeric Expressions		W			D
5505	Numeric Expressions		W			D
5510	Numeric Expressions		W			D
5515	Numeric Expressions		W			D
5520	Numeric Expressions		W			D
5525	Numeric Expressions		W			D
5530	Numeric Expressions		W			D
5535	Numeric Expressions		W			D
5540	Numeric Expressions		W			D
5545	Numeric Expressions		W			D
5800	Simple Comparisons		S			
5805	Simple Comparisons		S			
5810	Simple Comparisons		S			
5815	Simple Comparisons		W			D
5820	Simple Comparisons		S			
5825	Simple Comparisons		S			
5830	Simple Comparisons		S			
5835	Simple Comparisons		S			
5840	Simple Comparisons		S			
6000	Chained Comparisons		N	S		
6005	Chained Comparisons		N	S		
6010	Chained Comparisons		C	S		
6015	Chained Comparisons		W			D
6020	Chained Comparisons		C	S		
6025	Chained Comparisons		C	S		
6030	Chained Comparisons		N	S		

Table III (Continued)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
6085	Boolean Expression	v	N	N		v
6090	Boolean Expression	v	C	M	v	
6095	Boolean Expression	v	C	M	v	
6100	Boolean Expression		C	S		
6105	Boolean Expression		C	N		v
6100	Boolean Expression		C	S		
6115	Boolean Expression		N	S		
6120	Boolean Expression		N	S		
6125	Boolean Expression	v	N	M	v	
6130	Boolean Expression		N	S		
6135	Boolean Expression		C	F		v
6200	Assignment		W		v	D
6400	Exchange		C	S		
6405	Exchange		C	S		
6410	Exchange		C	S		
6415	Exchange		W			D
6420	Exchange		C	S		
6425	Exchange		C	S		
6430	Exchange		C	S		
6435	Exchange		C	F		v
6440	Exchange		C	S		
6445	Exchange		C	S		
6450	Exchange		W			D
6455	Exchange		C	S		
6460	Exchange		C	S		
6465	Exchange		C	S		
6470	Exchange		C	S		
6475	Exchange		C	S		
6500	DEFINE		C	S		
6600	LIKE		C	S		

Table III (Concluded)

JCVS Test Module Validation Summary

Module No.	Test Name	Modified by MITRE	GE-600 JOVIAL	IBM/360 JOVIAL	Complies by Analysis	Status not Resolved
6605	LIKE			S		
6610	LIKE		C	S		
6700	Overlays		C	S		
6705	Overlays		C	S		
6710	Overlays		C	F		v
6715	Overlays		C	S		
6720	Overlays		C	S		
6725	Overlays		C	F		v
6730	Overlays		C	S		
6735	Overlays	v	C	M	v	
6740	Overlays		C	F		v
6745	Overlays	v	C	M	v	
6750	Overlays		C	S		
6755	Overlays		C	S		
6760	Overlays		C	S		
6765	Overlays		C	S		
6770	Overlays		C	N		v
6775	Overlays		C	S		
6780	Overlays		C	N		v
6785	Overlays		W			D
6790	Overlays		C	N		v
6795	Overlays		C	N		v
6800	Overlays		C	S		
6805	Overlays		N	N		v
9998	Module 9998 *	v	M			

REFERENCES

1. Department of the Air Force, Air Force Manual AFM 100-24: Standard Computer Programming Language for Air Force Command and Control Systems. Short Title: CED2400, Washington, D.C., 15 June 1967.

2. Hq. Electronic Systems Division (AFSC), ESD-TR-70-278: User's Manual JOVIAL Compiler Validation System, Directorate of Systems Design and Development, L. G. Hanscom Field, July 1970.

3. IBM Corporation, Program Information Department, Contributed Program Library Type III, 360-03.2.010: The System/360 JOVIAL Compiler, Hawthorne, N. Y.

4. General Electric Co., Information Systems, CPB-1650: GE-600 Line JOVIAL, Phoenix, Arizona, 1970.

5. Frank Engel, Jr., The Air Force COBOL Compiler Validation System (CCVS), The MITRE Corporation, ESD-TR-71-61 (MTR-1953). Contract F19(628)-71-C-0002, Bedford, Mass., 28 September 1970.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

The MITRE Corporation
P. O. Box 208 Bedford, Massachusetts

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

THE AIR FORCE JOVIAL COMPILER VALIDATION SYSTEM (JCVS)

4. OESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Frank Engel, Jr.

6. REPORT OATE

AUGUST 1971

7a. TOTAL NO. OF PAGES

43

7b. NO. OF REFS

5

8a. CONTRACT OR GRANT NO.

F19(628)-71-C-0002

b. PROJECT NO.

8510

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

ESD-TR-71-236

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

MTR-2091

10. OISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Electronic Systems Division, Air Force
Systems Command, L. G. Hanscom Field,
Bedford, Massachusetts 01730

13. ABSTRACT

The Air Force JOVIAL Compiler Validation System (JCVS) was developed to assist in the validation of performance of proposed JOVIAL language processors. This report discusses the JCVS in the context of its use by the Air Force Directorate of ADPE Selection. It provides a certification of the audit test modules, and makes recommendations for improvements to the JOVIAL audit capability. It is recommended that the audit programs be used independently of the JCV System.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
AIR FORCE PROCUREMENT						
COMPILERS						
COMPUTER PROGRAMS						
COMPUTERS						
COMPUTER SYSTEMS PROGRAMS						
JOVIAL						
PROGRAMMING LANGUAGES						